In the C language, you use the `printf` statement to display formatted output. You can also use it in the C++ language. But, you should use the `cout` object with stream manipulators to format output in C++, rather than the `printf` statements.

You may still see the `printf` statements in some C++ programs. The `printf` statement formats the output using following syntax:

```c
printf(format, item1, item2, ..., itemk)
```

where `format` is a string that may consist of substrings and format specifiers. A format specifier specifies how an item should be displayed. An item may be a numeric value, a character, a boolean value, or a string. Each specifier begins with a percent sign. Table 1 lists some frequently used specifiers:

### Table 1

#### Frequently Used Specifiers

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Output Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%b</code></td>
<td>a boolean value</td>
<td>true or false</td>
</tr>
<tr>
<td><code>%c</code></td>
<td>a character</td>
<td>'a'</td>
</tr>
<tr>
<td><code>%d</code></td>
<td>a decimal integer</td>
<td>200</td>
</tr>
<tr>
<td><code>%f</code></td>
<td>a floating-point number</td>
<td>45.460000</td>
</tr>
<tr>
<td><code>%e</code></td>
<td>a number in standard scientific notation</td>
<td>4.556000e+01</td>
</tr>
<tr>
<td><code>%s</code></td>
<td>a string</td>
<td>&quot;Java is cool&quot;</td>
</tr>
</tbody>
</table>

Here is an example:

```c
int count = 5;
double amount = 45.56;

printf("count is %d and amount is %f", count, amount);
```

```
display            count is 5 and amount is 45.560000
```

Items must match the specifiers in order, in number, and in exact type. For example, the specifier for `count` is `%d` and for `amount` is `%f`. By default, a floating-point value is displayed with six digits after the decimal point. You can specify the width and precision in a specifier, as shown in the examples in Table 2.

### Table 2

#### Examples of Specifying Width and Precision

<table>
<thead>
<tr>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%5c</code></td>
<td>Output the character and add four spaces before the character item.</td>
</tr>
</tbody>
</table>
\%6b  Output the boolean value and add one space before the false value and two spaces before the true value.

\%5d  Output the integer item with width at least 5. If the number of digits in the item is < 5, add spaces before the number. If the number of digits in the item is > 5, the width is automatically increased.

\%10.2f  Output the floating-point item with width at least 10 including a decimal point and two digits after the point. Thus there are 7 digits allocated before the decimal point. If the number of digits before the decimal in the item is < 7, add spaces before the number. If the number of digits before the decimal in the item is > 7, the width is automatically increased.

\%10.2e  Output the floating-point item with width at least 10 including a decimal point, two digits after the point and the exponent part. If the displayed number in scientific notation has width less than 10, add spaces before the number.

\%12s  Output the string with width at least 12 characters. If the string item has less than 12 characters, add spaces before the number. If the string item has more than 12 characters, the width is automatically increased.

\textbf{\textless Side Remark: left justify\textgreater}

You can put the minus sign (-) in the specifier to specify that the item is left-justified in the output within the specified field. For example, the following statement

\begin{verbatim}
printf("%8d%-8s\n", 1234, "Java");
printf("%-8d%-8s\n", 1234, "Java");
\end{verbatim}

displays

1234Java
1234Java

\textbf{CAUTION:} The items must match the specifiers in exact type. The item for the specifier \%f or \%e must be a floating-point type value such as 40.0, not 40. Thus an \textbf{int} variable cannot match \%f or \%e.

\textbf{TIP:} The \% sign denotes a specifier. To output a literal \% in the format string, use \\%. 